,

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Richard Učník

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Richard Učník**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Projektově.CZ s.r.o.
2. Struktura závěrečné zprávy:
a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c. Zvolený postup řešení zadaných úkolů
d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeňka Chmelíková, Ph.D.**

Konzultant bakalářské práce: Ing. Zdeněk Solnický

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * 1D16 * OSTRAVA *

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26 odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Chtěl bych poděkovat Ing. Zdeňku Solnickému a Richardovi Římanovi, za jejich trpělivost a cenné rady, které mi za dobu praxe strávené ve firmě Projektově.CZ předali.

Abstrakt

Cílem této bakalářské práce bylo seznámit se s vývojem moderních dynamických webových aplikací pomocí javascriptové knihovny ReactJS. Bakalářská práce byla vykonána formou praxe ve firmě Projektově.CZ, kde jsem se v největší míře zabýval implementací systému na měření stráveného času uživatelů nad úkoly a jeho integrací do softwaru Projektově.CZ. Dále jsem se podílel na vývoji mobilní aplikace, kterou by mohli využívat zákazníci, vlastnící přístroje s operačním systémem iOS i Android.

Klíčová slova: ReactJS, Projektově.CZ, iOS, Android

Abstract

The main goal of this thesis was to develop modern dynamic web application with ReactJS. Bachelor‘s work was done by individual professional practice in company Projektově.CZ, where I spent most of my time implementing system which can track users time they spent on their tasks and then integrate this system into Projektově.CZ. Then I was part of a team which developed mobile application that can be used by both iOS and Android users.

Key Words: ReactJS, Projektově.CZ, iOS, Android

Obsah

Seznam použitých zkratek a symbolů

DOM	–	Document Object Model
JS	–	JavaScript
SaaS	–	Software as a Service
XML	–	eXtensible Markup Language
HTML	–	HyperText Markup Language
API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
SASS	–	Syntactically Awesome StyleSheets

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Čas jsou peníze. Mnoho firem si tento fakt uvědomuje a proto neváhají investovat do softwaru, který by jim pomohl jejich projektové řízení zjednodušit, zefektivnit. Vývojem takového softwaru se zabývá tým ve firmě Projektově.CZ a součástí tohoto týmu jsem se na pár měsíců stal i já.

Mým cílem bylo implementovat webovou aplikaci, ve které by si mohl uživatel jednoduše a přehledně spravovat svůj čas strávený nad jednotlivými úkoly, které mu jsou zadávány v hlavní aplikaci Projektově.CZ. Tím, že se zpřehlední nad čím vlastně zaměstnanci ve firmě pracují, napomáhá zefektivnit firemní řízení a proto je tato funkce nutnou součástí dobrého softwaru pro projektové řízení.

Problém s měřením času nad úkoly mají hlavně marketingové firmy a grafická studia. Jelikož mají velké množství velmi specifických, drobnějších úkolů, které tvoří pomyslný strom projektu. Tyto firmy potřebují velmi efektivní, přehledný software, umožňující sledovat náklady lidských zdrojů na projektu.

2 Projektově.CZ

2.1 Informace o společnosti

Společnost Projektově.CZ s.r.o. vznikla 11.2.2013 na základě technologického transferu ze společnosti Railsformers s.r.o., která se od roku 2010 věnuje vývoji aplikací ve frameworku Ruby On Rails. Hlavními aktivitami společnosti Projektově.CZ je poskytování služby Projektově.CZ – programu pro snadné projektové řízení.

Projektově.CZ je službou cílenou na malé a střední firmy, které řeší otázku efektivního úkolování. Jak už název společnosti naznačuje, problematiku řeší s pomocí projektového řízení. Cílem společnosti je nabízet software na internetu jako službu, která umožní uživateli získat přehled nad pracovními úkoly - ať už svými tak i svých kolegů.

2.2 Popis služby Projektove.CZ

Podstatou Projektově.CZ je poskytovat software jako službu, zkratkou SaaS (Software as a Service). SaaS je poskytování softwaru přes internet, kdy klient není nucen instalovat program lokálně na počítači. Ke spuštění programu mu stačí internetový prohlížeč a přístup na internet. O provoz, aktualizaci a údržbu systému se stará společnost dodávající SaaS.

Služba Projektově.CZ se drží trendu moderních programů a je proto vytvořen vysoký tlak na jednoduchost užívání, přehlednost rozhraní. Služba je tvořena pro uživatele, tudíž v první fázi je intenzivně komunikována s prvními uživateli, pro jejich potřeby jsou vytvářeny speciální pohledy na data, výstupy.

2.3 Přínos pro koncové uživatele

1. Krátká implementační doba - klient poptávající naše řešení může službu Projektově.CZ používat už nadcházející den.

2. Neustálá dostupnost - na jakémkoliv zařízení připojeném do internetu - tj. dostupnost z práce, z domova, z počítače, tabletu i mobilního telefonu.

3. Platba jen za to, co společnost skutečně využije - rozlišení v ceníku podle poptávané podpory a poptávaného diskového prostoru.

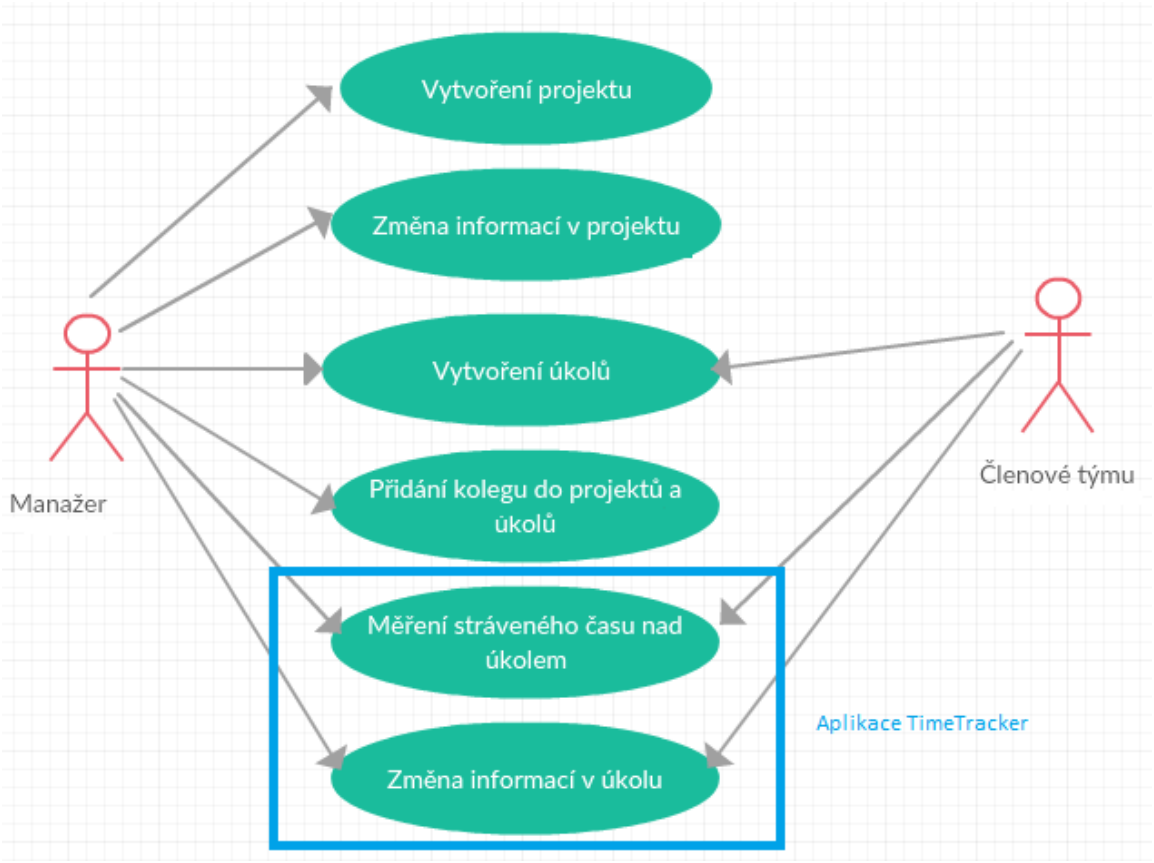
4. Snížení nákladů a nároků na IT oddělení - provoz i podporu kompletně zajišťuje Projektově.CZ.

5. Aktualizace bez starostí – aktualizace a správu programu provádí Projektově.CZ.

2.4 Use case diagram Projektově.CZ

Use case diagram reprezentuje interakce uživatelů v systému i funkce systému samotného. Měl by být navržen tak, aby mu zákazník jednoduše porozuměl a tím ho seznámil se základními funkcemi softwaru. [1]

Na obrázku 1 je znázorněn diagram základních funkcí Projektově.CZ, část kterou jsem měl za úkol implementovat je vyznačená modře.



Obrázek 1: Use case diagram Projektově.CZ

3 Teoretický základ potřebný k práci ve firmě Projektově.CZ

3.1 Projektové řízení

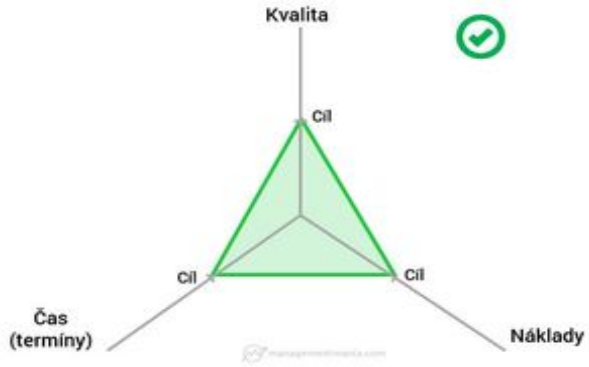
Projektové řízení slouží k naplánování a pozdější realizaci úkolů, které je potřeba uskutečnit v požadovaném termínu s předem určenými náklady tak, aby uživatel dosáhl svých stanovených cílů. Jde tedy o účinné, efektivní plnění zadaných úkolů.

Hlavním objektem v projektovém řízení je projekt, který představuje množinu úkolů, které je potřeba naplánovat a provést, aby bylo dosaženo požadovaného pokroku.

Cílem projektového řízení je tedy naplánovat kroky, které by efektivně vedly k realizaci úspěšného projektu, kdy v předem daném čase, s určenými prostředky bude dosaženo požadovaného cíle. [6]

3.1.1 Trojimperativ projektového řízení

Trojúhelník projektového řízení, také Magický trojúhelník projektového řízení nebo Projektový trojimperativ je vyjádření tří základních parametrů, kterými je měřen úspěch projektu, tedy čas, rozpočet projektu a kvalita výstupů. Na obrázku 2 je znázorněn trojimperativ projektového řízení. [7]



Obrázek 2: Projektový trojimperativ, ukázka úspěšného projektu

3.1.2 Řízení projektů v praxi

Řízení projektů přináší v praxi obvykle různé komplikace, týkající se i projektů dobře naplánovaných. V praxi pak dochází k porušení jednoho ze zmiňovaných parametrů. Nejčastěji dochází ke zpoždění termínu, k překročení nákladů, někdy se při snaze dodržet tyto dva faktory zhorší kvalita výstupů. Každá z těchto situací je pro zákazníka negativem - pozdě dodaný výstup projektu, přestože je kvalitní a za původní cenu může způsobit stejné problémy jako nekvalitní výstup přesto, že je dodaný včas. [6]

3.1.3 Projektové řízení v ČR

Momentálně se na českém trhu objevuje vícero firem, které se zabývají vývojem softwaru, jež by usnadňoval projektové řízení. České firmy si začínají uvědomovat, že k efektivnější realizaci projektů potřebují zlepšit svá interní projektová řízení a proto neváhají investovat do systému, který by jim s tímto problémem vypomohl.

3.2 ReactJS

3.2.1 Co je React?

ReactJS je JavaScriptová knihovna určena pro vývoj dynamických webových aplikací. Je vyvýjena společností Facebook a byla zpřístupněna veřejnosti v roce 2013. React umožňuje tvorbu komplexních webových aplikací, ve kterých se data mění bez nutnosti aktualizace stránky (Page Refresh). Hlavním cílem Reactu je vytvářet rychlé, škálovatelné aplikace, nenáročné pro vývoj. [8]

Obrázek 3: Graf populárních technologií z Google Trends

3.2.2 Virtual DOM

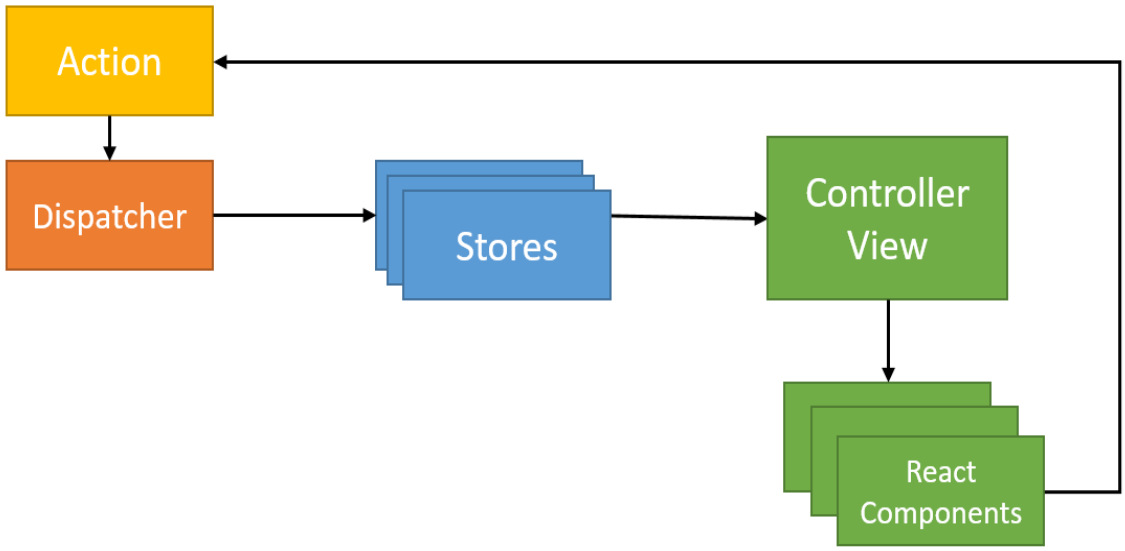
DOM je zkratka z anglického Document Object Model a je to objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je API umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí.

Výhodou Reactu je to, že pouze deklarativně definujeme strukturu HTML. React si z toho poskládá svůj vlastní virtualní DOM a pak ho porovnává se skutečným DOMem a rozdíly velmi efektivně aktualizuje.

Například u jQuery, který využívá imperativní přístup, může nastat problém u rozsáhlých projektů z důvodů narůstající složitosti spojené s prací s jednotlivými DOM elementy. [8]

3.3 Flux

Flux je architektura, která umožňuje vytvářet komplexní aplikace. Princip spočívá v jednosměrném toku dat. V kombinaci s knihovnou Immutable.js a tím prací s immutable objekty mi tento přístup umožnil zefektivnit, zjednodušit práci s daty v aplikaci. Na obrázku 4 je zobrazeno schéma architektury flux. [9]



Obrázek 4: Koncept jednosměrného toku dat v aplikacích

4 Aplikace TimeTracker

4.1 Nastínění problému

V rámci hlavní aplikace Projektově.CZ se vedení firmy setkalo s problémem, kdy jim chyběla jednoduchá, intuitivní funkce jak měřit uživatelům čas strávený nad jejich zadanými úkoly. To je pro manažery projektů nepříjemnost, protože nemají přehled nad prací svých kolegů, čímž je snížena jak efektivita týmu, tak celé firmy.

S tímto problémem se potýkají hlavně firmy, které prodávají svůj čas, například poskytnutím nějaké služby svým zákazníkům. Tyto firmy si potřebují dělat přesné záznamy o svých úkolech. Proto potřebují efektivní, intuitivní, spolehlivý software, který by jim na pár kliknutí umožnil tyto úkoly měnit a zpřehlednit tak práci v rámci firmy.

Aplikace by měla být nezávislá na hlavní aplikaci Projektově.CZ, musí mít implementovanou vlastní přihlašovací stránku. Uživatel se bude moci přesměrovat na aplikaci i z hlavní aplikace, a to tlačítkem v detailu daného úkolu.

Tato aplikace by se tímto měla stát hlavním prvkem ke správě uživatelova času v systému, který poskytuje Projektově.CZ. Pokud zaměstnanec ví na čem má pracovat, nemusí se vůbec přihlašovat do aplikace hlavní, stačí mu na to TimeTracker.

4.2 Základní funkce aplikace

Před samotným začátkem vývoje webové aplikace muselo vedení firmy specifikovat co se od ní očekává, jaké jsou základní, nezbytné funkce a technologické nároky. S vývojem aplikace se pár požadavků částečně změnilo. Momentálně je v produkci nasazena verze aplikace, splňující všechny následující body zadané a upravené vedením za dobu vývoje.

1. Aplikace by měla být nezávislá na hlavní aplikaci Projektově.CZ, bude mít svůj vlastní login stránku, tím pádem k ní uživatel bude moci přistoupit i bez nutnosti přihlášení do hlavního systému.

2. V aplikaci bude využita architektura Flux. Implementována bude s využitím knihovny Alt.js, v aplikaci se manipuluje s daty a proto je využití tohoto přístupu výhodou.

3. Když si uživatel spustí stopky nad úkolem a odhlásí se nebo vypne prohlížeč, při následovném přístupu do aplikace by se měla být schopna obnovit a pokračovat v již započatém sledování času.

4. Uživatel si bude moci v aplikaci změnit u naměřeného záznamu k jakému úkolu se čas vztahoval, datum i dobu, taktéž si k záznamu může připsat komentář.

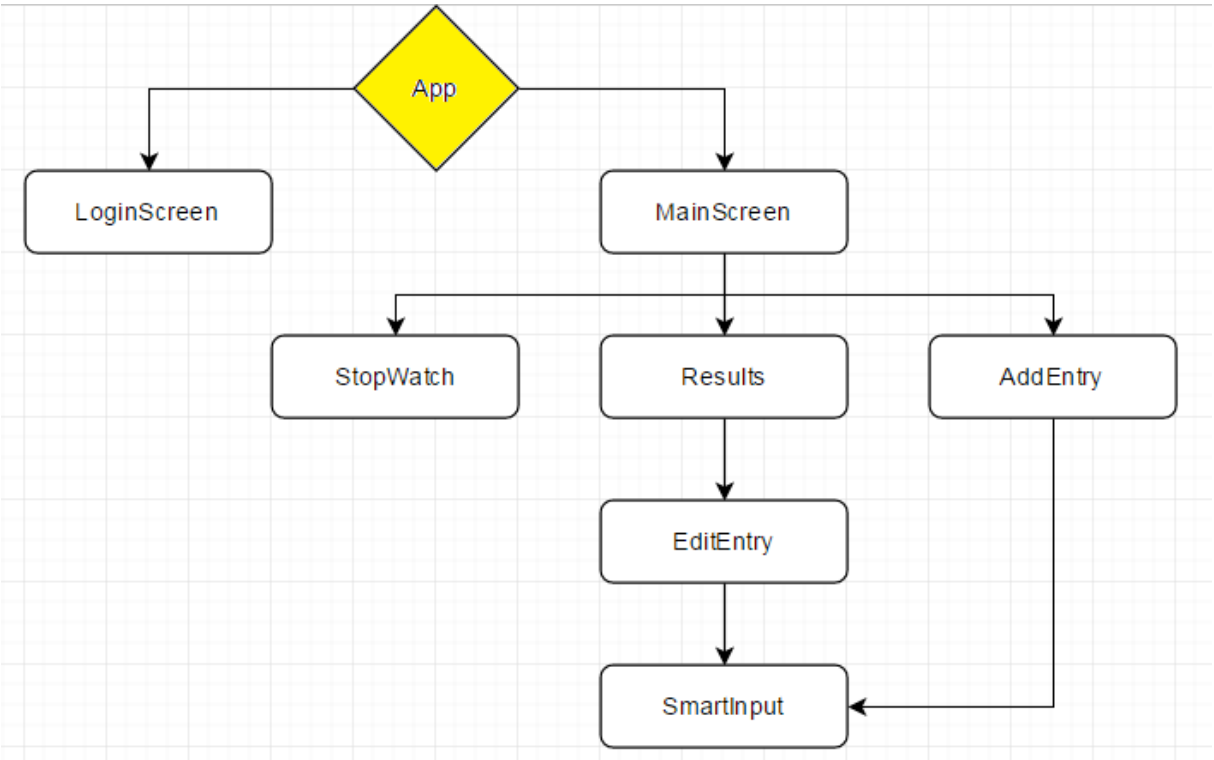
5. Uživatel si bude moci vytvořit záznam, aniž by musel spoustět stopky nad úkolem.

6. Aplikace bude nastylována pomocí konceptu Material design.

5 Postup řešení při vývoji aplikace

5.1 Návrh aplikace

Aplikace by měla mít co možná nejjednodušší, intuitivní uživatelské rozhraní. Hlavními komponentami aplikace bude přihlašovací stránka, hlavní stránka, která se bude skládat z vícero částí a vyskakující okna, které budou sloužit k úpravě časového záznamu nebo k tvorbě záznamu nového.



Obrázek 5: Zjednodušený strom projektu

Aplikaci jsem navrhnul tak, že v kořeni stromu projektu bude řidící komponenta, která bude rozhodovat zda-li se má vykreslit komponenta přihlašovací stránky, či stránka hlavní. V komponentě App se tedy nachází funkce na kontrolu uživatelských udajů, pokud chybí vykreslí se přihlašovací stránka. V komponentě, která odpovídá za vykreslení přihlašovací stránky jsou funkce pro komunikaci se serverem. Při uspěšném přihlášení se uživatelovy informace zašlou callback funkcí zpět do komponenty App, čímž dojde ke změně stavu komponenty, tím dojde k překreslení a následně pak k přihlášení. Strom projektu je znázorněn na obrázku 5.

Hlavní stránka aplikace se skládá ze třech částí.

1. Komponenta obsahující logiku pro měření stráveného času
2. Komponenta, která se stará o prezentaci uživatelových úkolů, kde si pak vybere úkol nad kterým chce pracovat
3. Komponenta prezentující záznamy naměřené uživatelem

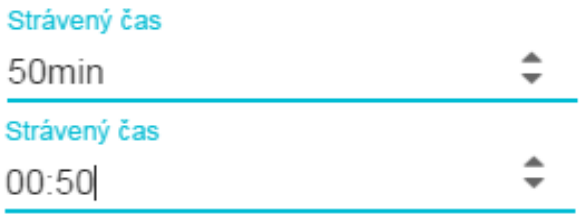
5.2 Komponenta SmartInput

Při implementaci komponent jsem se řídil několika pravidly, které jsou v prostředí Reactu žádané.

1. Komponenty by měly obsahovat pouze jednu logiku, měly by být jednoúčelné
2. Měly by být implementovány tak, aby byly univerzální, to znamená, že je můžeme využít na více místech v projektu

Jednu takovou komponentu představím blíže a jedná se o kontrolovaný input, který parsuje uživatelův vstup. Ten pak funkce v komponentě přetvoří do tvaru kompatibilního s tvarem, který vyžaduje server. Očekávaným vstupem uživatele je časový údaj, v tomto případě je jedno v jakém tvaru je do vstupu vložen, funkce v komponentě by si s ním měly poradit. Tvary, které dokáže komponenta přijmout.

1. 1,5
2. 1.5
3. 1,5hod
4. 120min
5. 120
6. 1:30



Obrázek 6: Ukázka inputu - 50min je uživatelův vstup, po stisku tlačítka enter se vtup automaticky změní do formátu 00:50

Uživatel má taktéž možnost manipulovat s časem pomocí šipek, kolečkem na myši nebo klávesnicí, pomocí šipek nahoru, dolů.

5.2.1 Důvod k implementaci SmartInputu

Tuto komponentu jsem implementoval z důvodu uživatelského pohodlí, jelikož celá tato aplikace pracuje s časovými záznamy, je tato funkce užitečným přínosem pro zjednodušení práce s touto aplikací, což je základním požadavkem pro celou aplikaci, jednoduchost, intuitivita.

Tato komponenta se mému vedoucímu práce natolik zalíbila, že je momentalně zakomponována i v hlavní aplikaci Projektově.CZ a je využívána uživateli každý den.

5.3 Komunikace se serverem

Pro komunikaci se serverem jsem se rozhodl využít knihovnu třetí strany a to knihovnu *fetch* a *fetch polyfill*. Fetch polyfill proto, že i v dnešní době se můžeme setkat s uživateli, kteří používají staré prohlížeče. Takže aby i těmto uživatelům aplikace bez problému fungovala, je tato knihovna nutnou součástí aplikace.

5.3.1 Uživatelé s velkým množstvím úkolů

Tento problém jsem se rozhodl vyřešit pomocí metody *response chunking*. V podstatě se jedná o nastavení maximálního limitu pro odpověď ze strany serveru. Zjistím kolik takových chunků dat musím stáhnout a s touto informací následně stáhnu zbytek dat.

Výhodou tohoto přístupu je, že uživatel nestahuje jeden velký objekt s daty, na který by v extrémních případech musel dlouho čekat, například u uživatelů co mají obrovské množství úkolů. Požadavek na server se mu rozdělí na menší části, má tím pádem rychlejší přístup k části dat a to vede k plynulejšímu chodu aplikace. Ukázka části kódu, představující metodu response chunking, je vyzobrazena ve výpise 1.

```
fetch(`https://${address}/api/issues?scope=open&limit=${limit}&assignedToId=
    ${id}&key=${apiKey}`)
.then(checkStatus)
.then(parseJSON)
.then(json => {
   TaskActions.addTasks(json) //pridani ukolu do uloziste
   let chunks = Math.ceil(json.meta.totalCount / limit)
   if (chunks > 1) {
      for (let ch = 1; ch < chunks; ch++) {
         fetch(`https://${address}/api/issues?scope=open&limit=${limit}&
             offset=${limit * ch}&assignedToId=${id}&key=${apiKey}`)
         .then(checkStatus)
         .then(parseJSON)
         .then(json => TaskActions.addTasks(json))
      }
   }
})
```

Výpis 1: Ukázka funkce pro response chunking

5.4 Práce s daty v aplikaci

Cílem aplikace je spravovat data uživatele, proto je nutné zvolit prostředí, které umožňuje spolehlivou, předvídatelnou manipulaci s těmito daty, aby se informace neztratily, nepřepsaly, nesmazaly. Řešením je využití konceptu uložisť (storů). Dalším důležitým článkem v tomto řešení je použití knihovny Immutable.js, která mi umožní v aplikaci pracovat s immutable objekty.

5.4.1 Immutable objekty

Mutace jsou úkazem lokálního programování. Všechna data mají své místo a když dojde k jejich změně, tak je na jejich místě nahradíme. Immutabilní data mají tu vlastnost, že každá změna vytvoří data nová, změněná.

Obrovskou výhodou immutability v Reactu je to, že počíta změny, ke kterým došlo od předchozího stavu a na základě nich provede jen nutné změny v DOMu prohlížeče. Díky tomu je React efektivní a velmi rychlý. Při použití immutable objektů je výpočet rozdílů v datech otázkou porovnání referencí, případně hash struktur. Díky tomu je možné snížit dobu vykreslování o polovinu a v některých případech i na třetinu viz. obrázek 7. [9]

Mutable Object

```
> Perf.printWasted()
```

(index)	Owner > component	Wasted time (ms)
0	"SlowItems > SlowItem"	723.7059941980988

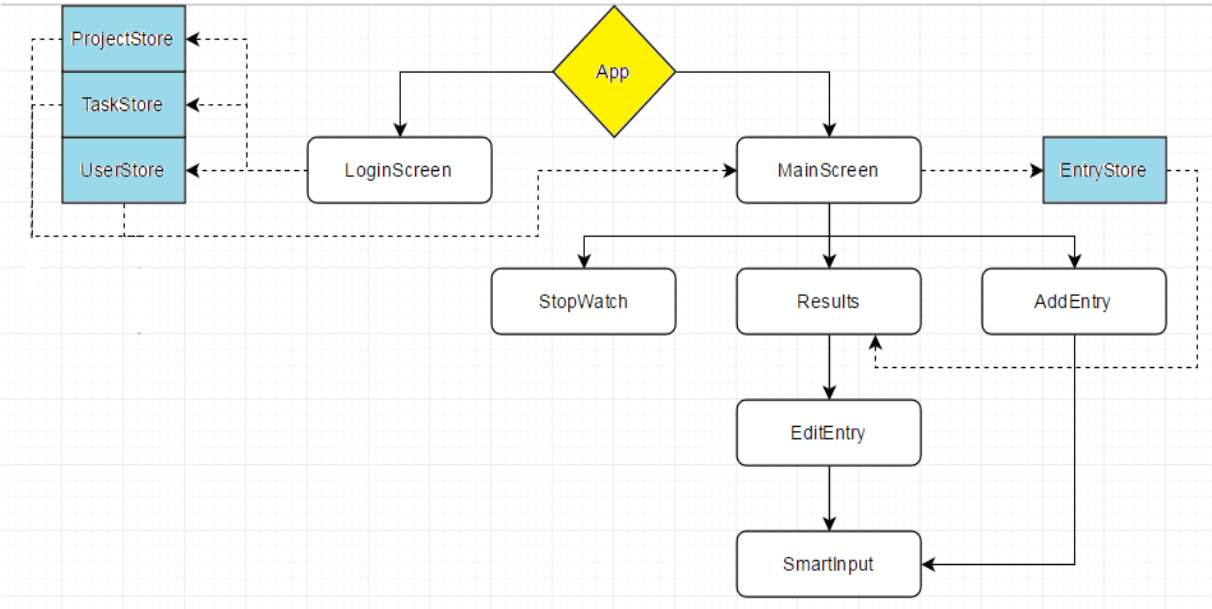
```
Total time: 967.63 ms
```

Immutable Object

```
> Perf.printWasted()
▶ Array[0]
Total time: 232.40 ms
```

Obrázek 7: Porovnání použití mutable a immutable datové struktury při testování s funkcí *shouldComponentUpdate*

5.4.2 Architektura flux v aplikaci



Obrázek 8: Ukázka použití konceptu flux v aplikaci, modrá barva představuje jednotlivá uložiště v aplikaci

5.4.3 Návrh architektury flux

Při použití tohoto konceptu bylo důležité si uvědomit, jaká uložiště v aplikaci vlastně využiji. V obrázku 8., kde je zobrazen návrh aplikace i implementovanými uložistěmi lze vidět, že jsem se nakonec rozhodl pro vytvoření čtyr uložisť, které mi udržují stav určitých dat v aplikaci.

Při procesu přihlašování, kdy na server posílam uživatelem vložené informace z login stránky, se mi ze serveru vrací objekt s informacemi o přihlášeném uživateli. Tyto informace, konkrétněji api klíč využiji k získání uživatelových projektů a úkolů. Všechny tyto data vkládám do uložišť, do kterých pak v případě potřeby přistupuji z jiných komponent. Jedná se o uložiště ProjectStore, TaskStore a UserStore.

Poslední uložiště, EntryStore slouží ke správě záznamů vytvořených v rámci aplikace.

5.4.4 Jak uložiště fungují?

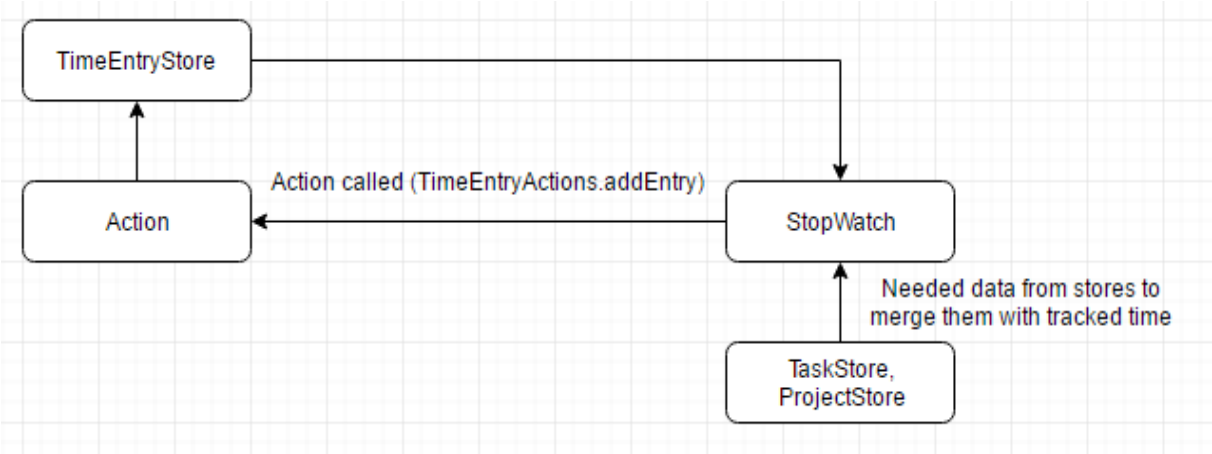
V tomto odstavci blíže popíši jak funguje koncept Flux v mé aplikaci, jako ukázku využiji obrázek 9. Tato ukázka se týka situace, kdy se uživatel rozhodne ukončit měření času nad úkolem. V ten moment se zavolá funkce, ve které se vyšle akce na uložiště, kde ukládám uživatelovy záznamy.

Nové data putují do uložiště, ve kterém se vytvoří nový aktualizovaný objekt s daty původního objektu a daty novými. Tím dojde ke změně stavu v uložisti.

Komponenty, které na toto uložiště naslouchají se dozví o změně stavu. To vede k překreslení komponenty. V mém případě, když se aktualizoval stav uložiště s uživatelovými záznamy, komponenta, která se stará o prezentaci těchto dat se dozví o změně a překreslí se. Ukázka volání akce ve výpise 2.

```
httpPost(`https://${userInfo.address}/issues/${selectedTask.taskId}/
    time_entries.json`,
   { data }}) //odeslani zaznamu na server
.then(json => TimeEntryActions.addEntry({ data }))
.catch(error => console.log(error))
```

Výpis 2: Příklad volání akce z komponenty

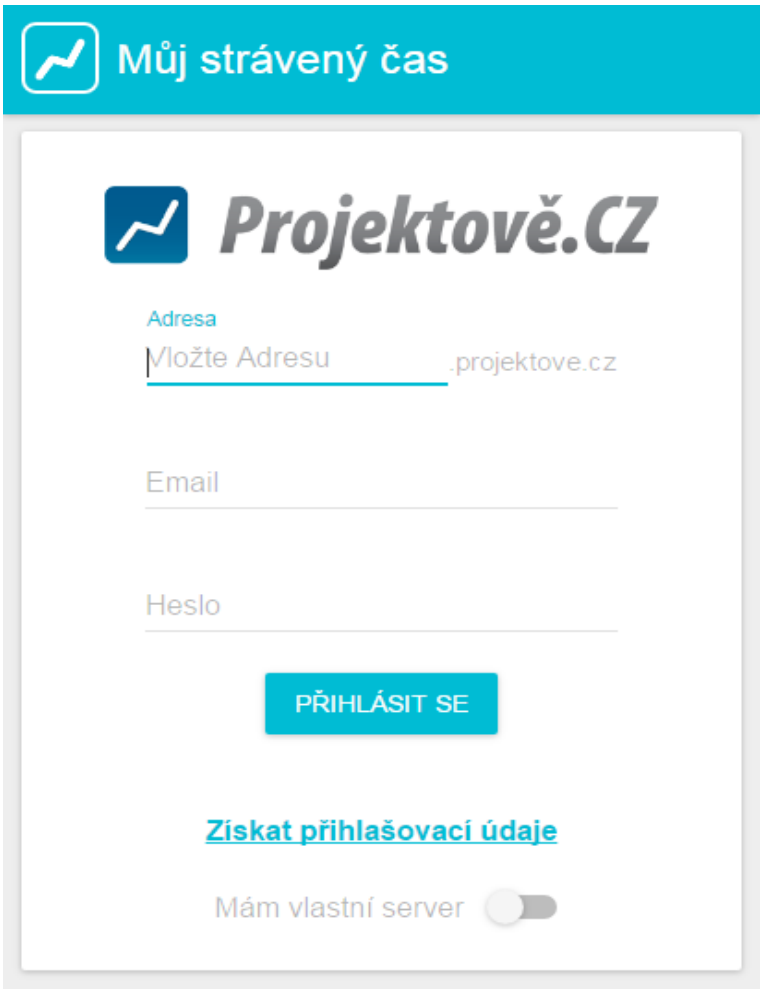


Obrázek 9: Ukázka použití konceptu flux v aplikaci

5.5 Material Design

Material Design je grafický koncept Googlu, který byl původně představen pro mobilní operační systém Android. Z mobilního prostředí postupně migruje do ostatních služeb Googlu. Material design prvků si můžeme všimnout například na stránce YouTube, ale i ve vyhledávači Google. [10] Cílem je:

1. Vytvořit prostředí, které dokáže propojit moderní design s intuitivním přehledným ovládaním prvků
2. Vytvořit prostředí, které umožní zformovat stejné postupy napříč platformami
3. Vstupní metody, jako je dotyk, myš a klávesnice jsou nejdůležitějším prvkem



Obrázek 10: Ukázka konceptu Material Design na přihlašovací stránce

5.5.1 SASS

SASS je kompilovaný jazyk rozšiřující syntaxi CSS o proměnné, cykly, podmínky, mixiny a funkce. Tento přístup pro stylování jsem zvolil z důvodu zjednodušení práce s CSS. SASS mi pomohlo zredukovat množství napsaného kódu, je přehlednější a lépe se udržuje. [11]

V aplikaci používám scss (Sassy CSS) soubory, které se musí zkompilovat do CSS pomocí SASS kompilátoru. Samotný scss soubor nelze použít, neboť prohlížeč nezná jeho syntaxi a proto nemůže nastylovat elementy.

5.6 GIT

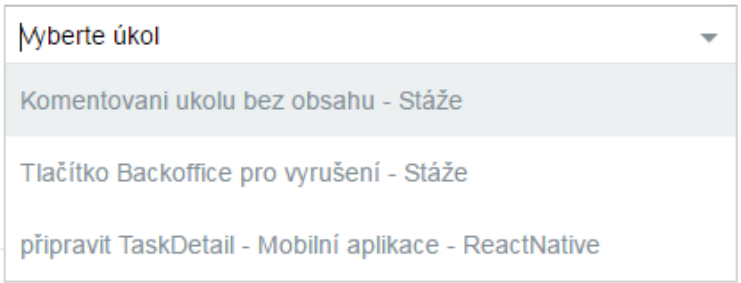
Mezi první věci, které jsem se musel ve firmě naučit, bylo správné používání systému pro správu verzí - git. Tento systém jednoduše umožňuje publikovat projekty ve verzích, usnadňuje vývoj aplikace funkcemi jako je například větvení, kdy do hlavní, master větve vkládáme kód, který je otestovaný a funguje. Pro vývoj nových funkcí vytváříme větve jiné, které později, až budou připravené, otestované můžeme propojit s větví hlavní a tim jednoduše software rozšiřovat. [12]

5.7 První aktualizace

V prvních dnech po uvedení aplikace do produkce se ukázalo, že někteří uživatelé mají problém s výběrem úkolu. Problém nastane v ten okamžik, kdy má uživatel úkoly pojmenované stejně a název se liší v projektu.

Proto bylo nutné propojit úkoly s projektem, ke kterému patří a poté tento objekt posílat do komponenty, ve které si uživatel vybírá úkol nad kterým chce pracovat.

Bylo nutné doimplementovat požadavek na server o stáhnutí uživatelových projektů, ty pak uložit do nově vytvořeného uložistě pro projekty. A v komponentě, která se stará o logiku vykreslování těchto informací naimplementovat funkce na propojení úkolu k danému projektu a tento objekt pak posílat do dané komponenty, která se stará o vykreslení tohoto objektu.



Obrázek 11: Uživateli požadovaná první aktualizace v aplikaci

5.8 Google Analytics

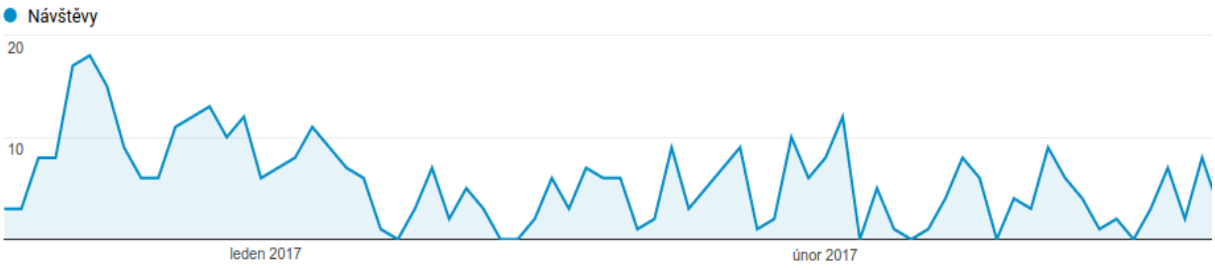
Google Analytics je nástroj od společnosti Google, který umožňuje vlastníkům webových stránek získávat statistická data o uživatelích svého webu. Díky této službě je možné sledovat aktuální návštěvnost i její historii, chování uživatelů a další. [13]

Na obrázku 12 si lze všimnout jakými zařízeními uživatelé přistupují do aplikace, v mém případě jsou to až na výjimky uživatelé stolních počítačů či notebooků. Počet nových uživatelů dosáhl za období leden a únor počtu 159.

Kategorie zařízení	Akvizice		
	Návštěvy ↓	% nových návštěv	Noví uživatelé
	494 Podíl z celku v %: 100,00 % (494)	32,19 % Prům. pro výběr dat: 32,19 % (0,00 %)	159 Podíl z celku v %: 100,00 % (159)
1. desktop	**468** (94,74 %)	32,91 %	**154** (96,86 %)
2. tablet	**18** (3,64 %)	5,56 %	**1** (0,63 %)
3. mobile	**8** (1,62 %)	50,00 %	**4** (2,52 %)

Obrázek 12: Zařízení se kterými uživatelé přistupují do aplikace

Na obrázku 13 je zobrazen vývoj návštěv uživatelů za měsíce leden a únor. Nižší počet uživatelů lze odůvodnit tím, že tato funkce je specificky zaměřená na určitou skupinu zákazníku, která momentálně v Projektově.CZ není tak početná. Denní průměr uživatelů se pohybuje kolem 5-10. S narůstajícím zájmem o aplikaci Projektově.CZ, se očekává i zvýšení počtu uživatelů TimeTrackeru.



Obrázek 13: Graf vývoje návstěv uživatelů v rámci aplikace

6 Mobilní aplikace Projektově.CZ

Ve zbývající části mé odborné praxe jsem se podílel na vývoji mobilní aplikace. Důvodem k tvorbě nové mobilní aplikace byl zájem zákazníků o aplikaci, kterou by mohli používat i na přístrojích s operačním systémem iOS.

Proto jsme se rozhodli využít nabytých znalostí z ReactJS a pro implementaci aplikace využít technologii React Native, která nám umožní vyvýjet aplikaci za pomocí JavaScriptu. Výsledný produkt pak bude kompatibilní jak s iOS, tak s Androidem. Rozdíl v kódu, mezi Android a iOS verzí je shruba 8% z celkového kódu použitého k implementaci aplikace. [2]

6.1 Funkčnost aplikace

Mobilní aplikace by měla uživatelovy poskytnou základní funkce webové aplikace Projektově.CZ, které pak bude moci využívat pohodlně v telefonu. Bude si moci zjistit nad jakými úkoly by měl pracovat, může se podívat na detail těchto úkolů nebo si úkoly může vytvářet a uzavírat.

6.2 Navigation

Mým úkolem bylo implementovat logiku pro navigování v rámci mobilní aplikace. Využil jsem k tomu komponentu Navigation a objekt navigator, který pak posílám do dalších komponent pomocí *this.props*. Tento objekt mi pak ulehčuje manipulaci se scénami.

Do objektu navigator si podle toho, kam uživatel klikne vkládam scény a pomocí tohoto objektu pak komponenta Navagation tyto scény vykresluje (viz. výpis 3). V komponentě Navigation je deklarovaná první scéna, která se vykreslí po přihlášení a funkce na vykreslování, které se řídí podle objektu navigator. Taktéž je v této komponentě implementována logika pro funkci Drawer, který si pak uživatel může "vytáhnout" z levé části displaye ve všech scénách a usnadnit si tím práci s aplikací. V Draweru jsou informace o uživatelových projektech a úkolech, do kterých se tímto může snadno dostat, nachází se tam také tlačítko pro nastavení aplikace. [15]

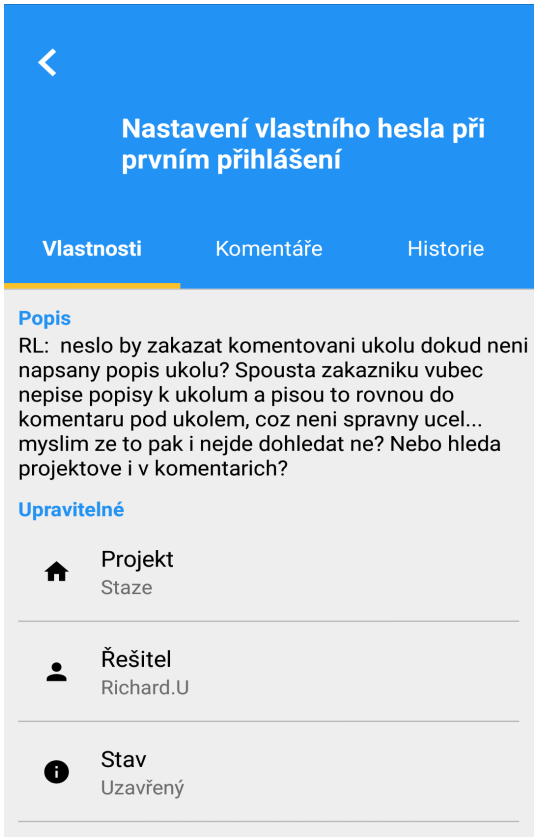
```
navigation(task, sceneName) {
   this.props.navigator.push({
      name: sceneName,
      task: task
   })
}
```

Výpis 3: Ukázka funkce použité pro vkládání scén do objektu navigator

6.3 Komponenta pro detail úkolu

Poté jsem měl za úkol implementovat komponentu pro detail úkolu, ve které by měl uživatel všechny potřebné informace, důležité pro splnění daného problému. Komponenty v React Native fungují obdobně jako v Reactu, takže jsem využil nabytých znalostí z vývoje webové aplikace.

Scéna s detailem úkolu se objeví po uživatelově interakci s výpisem úkolu. Zavolá se funkce při které se do objektu navigator zapíše komponenta s detailem úkolu a taktéž příslušná data vztahující se k úkolu. Komponenta Navigation zjistí změnu stavu objektu navigator a překreslí se. Při stisku šipky v Navigation baru se z objektu navigator odstraní poslední scéna a to inicializuje proces překreslení, uživatel se tak vratí zpět do výpisu úkolů, což je v případě této mobilní aplikace scéna hlavní.



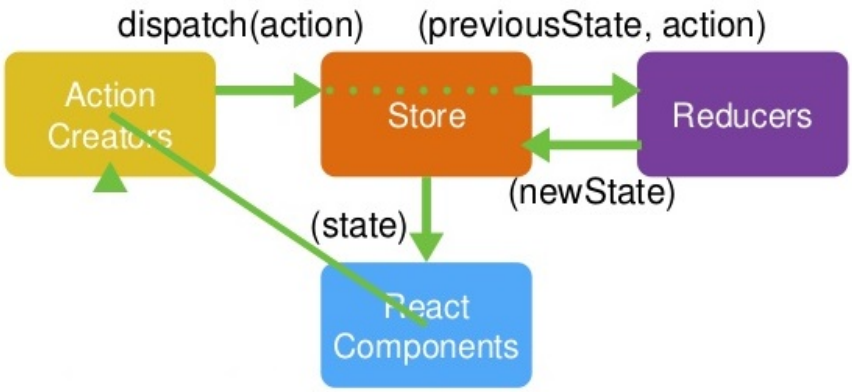
Obrázek 14: Komponenta detailu úkolu s implementovanou navigací

6.4 Redux

Redux je nejpopulárnější technologií, která se využívá při implementaci architektury Flux, jak v mobilních, tak webových aplikacích.

Na rozdíl od knihovny Alt.js má Redux velmi přehledně napsanou a udržovanou dokumentaci. Nepřehledná dokumentace je jedním z důvodů, proč Alt.js nemá tak početnou skupinu uživatelů.

Dalším rozdílem je jiný přístup při tvorbě uložisť, Redux využívá pouze jeden velký objekt, ve kterém skladuje všechny informace. Manipulaci s daty umožňují reducery. [16]



Obrázek 15: Ukázka popisující flow v Reduxu

6.4.1 Redux v mobilní aplikaci

Já jsem v aplikaci naimplementoval funkce, které po přihlášení ze serveru stáhnou uživatelovy projekty a úkoly (viz. výpis 4). V Reduxu existuje pouze jedno uložiště, je to objekt ve kterém jsou uložena všechna data a se kterými se manipuluje pomocí reducerů.

Princip volání akcí je stejný jako v případě knihovny Alt.js, tzn. pokud uživatel změní data v rámci aplikace, vyšlou se příslušné akce, které se v reducerech provedou a změní stav uložiště, tím dojde k překreslení komponent, které naslouchají na dané data v uložisti. Uživateli se tak zobrazí změna v uživatelském rozhraní.

```
export async function fetchProjects(): Action {
   try {
      let response = await httpGet(url, {
         'Accept':              'application/json',
         'Content-Type':        'application/json',
         'X-API-Authorization': global.store.getState().user.user.apiKey
```

```
      })
      return {
        type: 'projects/received',
        response: response
      }
    } catch (error) {
      return {
        type: 'projects/fetchingFailure',
        error
      }
    }
}
```

Výpis 4: Ukázka akce pro stáhnutí projektů ze serveru

7 Závěr

Hlavním cílem této práce bylo implementovat webovou aplikaci pro sledování stráveného času uživatelů nad jejich úkoly. Taktéž jsem se měl seznámit s prostředím webového vývoje a projektového řízení. Získal jsem zkušenosti s vývojem aplikací na internetu, od prázdného projektu až po plně funkční aplikaci připravenou pro produkci.

Jelikož mi implementace webové aplikace nezabarala celkovou délku praxe. Ve zbytku, shruba 120 hodin, jsem se podílel na vývoji mobilní aplikace Projektově.CZ, ve které jsem implementoval komponenty pro navigaci a detail úkolu, a zabýval se návrhem uložišť pro architekturu flux.

Vývoj obou aplikací byl přizpůsoben moderním trendům, byly využity koncepty, které jsou populární při vývoji moderních webových a mobilních aplikací. Práce s těmito mederními technologiemi byla velice zajímavá. V průběhu praxe jsem získal mnoho cenných zkušeností, jak z prostředí webového a mobilního vývoje, tak i zkušenosti s prací v týmu.

7.1 Další vývoj aplikace TimeTracker

Pro aplikaci se chystá nová funkce, tím bude BackOffice. Uživatelé zmíněných marketingových firem a grafických studií požadují funkci, při které by mohli v průběhu měření času nad úkolem stisknout tlačítko, které by představovalo jakési pozastavení práce. Čas, který by strávili v tomto přerušení by se jim taktéž zaznamenával. Může jít o nějaký telofonní hovor, či jiné vyrušení související s prací. Toto funkce by představovala další zpřesnění údajů naměřených uživateli aplikace.

Literatura

[1] Use Case Diagrams [online]. [cit. 2017-04-26]. Dostupné z: http://www.uml-diagrams.org/use-case-diagrams.html

[2] MASIELLO, Eric a Jacob FRIEDMANN. Mastering React Native. Packt, 2017. ISBN 9781785885785.

[3] EISENMAN, Bonnie. Learning React Native: building mobile applications with JavaScript. ISBN 9781491929001.

[4] CROCKFORD, Douglas. JavaScript: The Good Parts. O'Reilly, 2008. ISBN 9788184045222.

[5] FLANAGAN, David. JavaScript the definitive guide. 3rd ed. Beijing: O'Reilly, 1998. ISBN 9781565929586.

[6] Projektové řízení: Projekt, Projektové řízení [online]. [cit. 2017-04-22]. Dostupné z: http://navolnenoze.cz/blog/projektove-rizeni/

[7] Trojimperativ projektu [online]. [cit. 2017-04-22]. Dostupné z: https://support.office.com/cs-cz/article/Projektový-trojúhelník-8c892e06-d761-4d40-8e1f-17b33fdcf810

[8] ReactJS: ReactJS, VDOM, dokumentace [online]. [cit. 2017-04-22]. Dostupné z: https://facebook.github.io/react/

[9] Flux, ImmutableJS [online]. [cit. 2017-04-22]. Dostupné z: https://blog.risingstack.com/the-react-js-way-flux-architecture-with-immutable-js/

[10] Material Design [online]. [cit. 2017-04-22]. Dostupné z: https://material.io/guidelines/

[11] SASS [online]. [cit. 2017-04-26]. Dostupné z: http://sass-lang.com/documentation

[12] GIT [online]. [cit. 2017-04-22]. Dostupné z: https://wiki.archlinux.org/index.php/git

[13] Google Analytics [online]. [cit. 2017-04-26]. Dostupné z: https://www.google.com/analytics

[14] React Native, Redux [online]. [cit. 2017-04-22]. Dostupné z: https://medium.com/@jonlebensold/getting-started-with-react-native-redux-2b01408c0053

[15] Navigation in React Native [online]. [cit. 2017-04-22]. Dostupné z: http://blog.paracode.com/2016/01/05/routing-and-navigation-in-react-native/

[16] Redux, Mobile Applications in React Native [online]. [cit. 2017-04-22]. Dostupné z: http://makeitopen.com/tutorials/building-the-f8-app/data/